SYSTEM AND METHOD FOR PROVIDING DISTRIBUTED STATIC TIMING ANALYSIS WITH MERGED RESULTS

Kayhan Küçükçakar

Steve Hollands

Brian Clerkin

Loa Mize

Qiuyang Wu

Subra Sripada

Andrew J. Seigel

BACKGROUND OF THE INVENTION

Field of the Invention

[0001] The present invention relates to static timing analysis and in particular to distributed static timing analysis with merged results.

Description of the Related Art

[0002] Ignoring timing violations during design implementation can result in designs that fail to meet performance specifications or even fail in silicon. Given tight market windows and multi-million dollar re-spin costs, these design failures can be financially devastating to semiconductor companies. Therefore, design engineers are increasingly demanding techniques to identify timing violations in a design. One such technique, called static timing analysis, attempts to exhaustively analyze all critical paths of a design.

[0003] A limited number of static timing analysis (STA) runs, e.g. to provide a minimum time analysis and a maximum time analysis, can be sufficient for large process geometries and/or simple designs. Figure 1A illustrates an exemplary run 100 in which an STA tool 102 receives a design (e.g. a gate-level description) and an STA set-up 101. STA tool 102 performs the

static timing analysis and generates results 103 indicating any timing violations as well as coverage of the static timing analysis. At this point, a user could review results 103 and then attempt to fix the conditions that could be causing the timing violations, if present.

[0004] Unfortunately, with process geometries reaching 130 nm and below, gate counts exceeding 30 million, and designs growing in logic and core complexity, identifying timing violations with standard static timing analysis is rapidly becoming one of the most challenging problems for design engineers. Specifically, multiple runs of static timing analysis are typically required to take into account different modes (e.g. test mode, normal operation mode, power-down mode, etc.) as well as corners (e.g. process parameters including minimum and maximum temperature, minimum and maximum voltage, etc.).

[0005] Figure 1B illustrates a flow wherein a user (e.g. a design engineer) 112 uses a design and a generic STA set-up 110 as well as potential set-ups 111 (indicating specific modes and corners available to user 112) to perform multiple runs 100A, 100B, and 100C. For example, in run 100A, STA tool 102A receives a design and an STA set-up 101A that indicates a specific mode and corner for analysis. STA tool 102A performs the static timing analysis and generates results 103A indicating any violating paths as well as coverage of the static timing analysis using that specific mode and corner. In run 100B, STA tool 102B considers design and STA set-up 101B (same design, but different mode and/or corner) to generate results 103B. Similarly, in run 100C, STA tool 102C considers design and STA set-up 101C (same design, but different mode and/or corner than those indicated in design and STA set-ups 101A and 101B) to generate results 103C. Note that STA tools 102A-102C are running independently. That is, STA tools 102A-102C could be

running on the same computer or on different computers and could be running at the same time or at different times.

[0006] As STA tools 102A-102C complete their analysis, user 112 can review the results of the static timing analysis and perform debugging 113, as necessary, to mitigate timing violations. Of importance, user 112 must manually perform this user analysis/debugging 113 on results 103A-103C. Unfortunately, results 103A-103C typically form large, complex files in which each critical path in the design must be individually extracted for path profiling. Moreover, performing a comprehensive static timing analysis using available modes and corners can take many runs, e.g. 100-200 runs. Therefore, managing and/or merging the results from these runs can be very complex and time consuming.

[0007] Therefore, a need arises for a system and method of efficiently managing multiple static timing analysis runs using multiple modes/corners.

SUMMARY OF THE INVENTION

[0008] To ensure that a design meets performance specifications and can be successfully implemented in silicon, a user can perform static timing analysis on the design. Static timing analysis attempts to exhaustively analyze all critical paths of a design. Currently, a user debugs the design by manually analyzing the results of static timing analysis. With ever decreasing geometries and ever increasing design complexity, identifying timing violations with standard static timing analysis can be very complex and time consuming.

[0009] In accordance with one feature of the invention, a static timing analysis tool can manage multiple runs having different parameters and automatically merge the results of these runs. Parameters can include, but are not limited to,

modes (e.g. test mode, normal operation mode, power-down mode, etc.) and corners (e.g. process parameters including minimum and maximum temperature, minimum and maximum voltage, etc.).

[0010] The STA tool can perform the runs either in parallel or in series. Advantageously, the STA tool can save the full timing analysis generated by each run and then extract information from these saved results to form merged results for the design. In one embodiment including a plurality of modes and corners, the multiple static timing analysis runs can share information, and the plurality of modes and corners can be automatically analyzed to determine shared information between parallel runs.

[0011] Merged results can provide different levels of analysis coverage. For example, the merged results can indicate whether the design has been exhaustively analyzed for a particular mode and/or corner. Additionally, the merged results can indicate percentage coverage, i.e. how much of the total chip has been analyzed. Furthermore, the merged results can indicate timing checks that were not performed.

[0012] The merged results can also provide path information at various levels of detail. For example, merged results can indicate, for each path, timing violations based on specific modes and/or corners. In another embodiment, the merged results can indicate for each path in the design a percentage of times that timing violations exist for all analyzed modes/corners. In yet another embodiment, the merged results can include detailed timing information regarding each path. Thus, the merged results can now provide a depth of path analysis that to date has not been commercially available in the industry.

[0013] The merged results can also allow selectable accessibility to information generated in the multiple runs. For example, merged results can be organized as a database,

wherein a user can access information at a desired level of detail. Certain desired information can be designated (by the user or by default) before the analysis, thereby accelerating the production of reports including such merged results.

[0014] In one embodiment, saving results from each run can include restoring the database and making additional queries. The additional queries can be made from one or more runs. Advantageously, each query can add additional results to the saved results of each run.

[0015] The merged results can also highlight propagation of timing changes/violations in the design. For example, changing the speed of one cell in the design could impact violations in a certain percentage of paths. Therefore, the merged results can allow a user to quickly "leverage up" information.

[0016] In one embodiment, a predetermined set of parameters can be modified after completing an initial multi-mode/multi-corner analysis. At this point, an analysis can be performed to provide a what-if capability, thereby driving design optimization.

[0017] Therefore, using distributed processing, static time analysis can be performed thoroughly and efficiently. Moreover, by merging the results of such processing and extracting desired information at various levels of detail, a user can quickly and intelligently make decisions in debugging a design.

BRIEF DESCRIPTION OF THE FIGURES

[0018] Figure 1A illustrates an exemplary run in which an STA tool receives a design and an STA set-up. During the run, the STA tool performs the static timing analysis and generates results indicating any violating paths as well as coverage of the static timing analysis.

[0019] Figure 1B illustrates a flow wherein multiple runs can be performed, each run specifying a different STA set-up to analyze a design. In this flow, the user manually analyzes and debugs the results from the runs.

[0020] Figure 2 illustrates a flow wherein an STA tool can receive a design and manage multiple parallel runs (each run being performed with a predetermined mode and corner) analyzing that design. In this flow, the STA tool can automatically analyze and/or merge the results from the runs.

[0021] Figure 3 illustrates a flow wherein an STA tool can receive a design and manage multiple runs in series, thereby allowing a subsequent run to benefit from information provided from a previous run. The STA tool can automatically analyze and organize the results from the serial runs.

DETAILED DESCRIPTION OF THE FIGURES

[0022] Advanced chip technologies and increasingly complex designs typically require many runs of static timing analysis to ensure that possible timing violations are found before chip layout. Each run can use a specific mode and/or corner. Exemplary modes include, but are not limited to test mode, normal operation mode, and power-down mode. Exemplary corners include, but are not limited to, process parameters such as minimum temperature, maximum temperature, minimum voltage, and maximum voltage.

[0023] In accordance with one feature of the invention, a static timing analysis (STA) tool can advantageously manage these runs and automatically merge the results of these runs. These merged results can be organized to facilitate user review, thereby simplifying debugging of the design. To accelerate the runs, the STA tool can manage the multiple runs in parallel.

Alternatively, to leverage information generated in a run, the STA tool can manage the multiple runs in series.

[0024] Figure 2 illustrates an exemplary flow in which the multiple runs are managed in parallel. In this flow, an STA tool 204 can receive a design and a generic STA set-up 209 as well as one or more potential set-ups 208 that specify modes and/or corners available for design analysis. STA tool 204 can then initiate multiple parallel runs using the available set-ups. Note that although three runs 200A, 200B, and 200C are shown, an actual design analysis could include hundreds of runs. Of importance, each run can be performed with a different mode and/or corner.

[0025] In accordance with this embodiment, one or more STA tools can be used in parallel to perform the runs. For example, an STA tool 202A can receive the design and an STA set-up 201A, perform the static timing analysis using a specified mode and corner on the design, and then save results 203A from that static timing analysis in a database. An STA tool 202B can receive the design and STA set-up 201B, perform the static timing analysis using another specified mode and corner on the design, and then save results 203B from that analysis. Similarly, STA tool 202C can receive the design and STA set-up 201C, perform the static timing analysis using the specified mode and corner on the design, and then save results 203C.

[0026] In this embodiment, STA tool 204 can schedule STA tools 202A, 202B, and 202C to perform their static timing analysis in parallel, thereby significantly improving overall run time. Advantageously, STA tool 204 (shown twice in Figure 2 to clarify the process flow) can automatically combine the results of those runs in merged results 205, thereby facilitating an efficient user review. In one embodiment, STA tool 204 can merge results 203A, 203B, and 203C after all static

timing analysis is complete. In another embodiment, STA tool 204 can merge the results as they become available, thereby providing an increasingly more accurate merged results database over time.

[0027] In one embodiment, saving results from each run 200A, 200B, and 200C can include restoring (i.e. reading) the merged results database and making additional queries. The additional queries can be made from one or more runs. Advantageously, each query can add additional results to saved results 203A, 200B, 200C of each run 200A, 200B, and 200C. In another embodiment, saved results 203A, 203B, and 203C can include intermediate results to support arbitrary queries. These intermediate results can include a predetermined set of parameters that can be advantageously used in creating additional results.

[0028] Exemplary saved results 203A, 203B, or 203C can include one or more of cell delays, net delays, transition times, a timing graph, a parasitic network, path reports, bottleneck reports, a noise bump height, a noise bump width, a noise peak time, aggressors, victims, noise rejection curves, noise slack, current density, application attributes, user attributes, cells, nets, an analysis coverage, profiling of endpoints, profiling of paths, modes, case analysis propagation, design corner description, slack, design cost, constraints, annotations, combinational loops, clock re-convergence points, arrival times, required times, a timing window, crosstalk delays, and operating conditions.

[0029] Note that in any mode certain parts of the design can be disabled. Saved results 203A-203C can therefore indicate what part(s) of the design have/have not been analyzed. STA tool 204 can quickly and accurately merge this complex coverage information from runs 200A-200C.

[0030] Therefore, in accordance with one feature of the invention, merged results 205 can advantageously provide different levels of analysis coverage. For example, merged results 205 can indicate whether the design has been exhaustively analyzed for a particular mode and/or corner. Merged results 205 can also indicate percentage coverage, i.e. how much of the total chip has been analyzed. Merged results 205 can also indicate any timing checks that were not performed.

[0031] In accordance with another feature of the invention, merged results 205 can advantageously provide path information at various levels of detail. For example, in one embodiment, merged results 205 can indicate, for each path, timing violations based on specific modes and/or corners. In another embodiment, merged results 205 can indicate for each path in the design a percentage of times that timing violations exist for all analyzed modes/corners. In yet another embodiment, merged results 205 can include detailed timing information regarding each path. Thus, merged results 205 can now provide a depth of analysis for each path that to date has not been commercially available in the industry.

[0032] In accordance with yet another feature of the invention, merged results 205 can advantageously allow user-selected accessibility to information generated in the multiple runs. For example, merged results 205 can be organized as a database, wherein a user can access information at a desired level of detail. For example, a user could query merged results 205 regarding violations for a specific path in normal operation mode under all corners. The user could also query merged results 205 regarding violations for all paths on the chip in all modes under a specific corner, e.g. a high temperature condition. Thus, merged results 205 can provide users with a selectable level of analysis.

[0033] In accordance with yet another feature of the invention, merged results 205 can advantageously highlight propagation of timing changes/violations in the design. For example, changing the speed of one cell in the design could impact violations in a certain percentage of paths. Using merged results 205, these paths can now be identified. Therefore, merged results 205 can advantageously allow a user to quickly "leverage up" information.

[0034] Exemplary merged results 205 can include one or more of cell delays, net delays, transition times, timing graph, parasitic network, path reports, bottleneck reports, a noise bump height, a noise bump width, a noise peak time, aggressors, victims, noise rejection curves, noise slack, current density, application attributes, user attributes, cells, nets, analysis coverage, profiling of endpoints, profiling of paths, modes, case analysis propagation, design corner description, slack, design cost, constraints, annotations, combinational loops, clock re-convergence points, arrival times, required times, a timing window, crosstalk delays, and operating conditions.

[0035] Currently, users must manually extract all of the above-described information from the text files provided by a standard STA tool (e.g. results 103A-103C in Figure 1B). This manual extraction is extremely time consuming and prone to error. Therefore, in contrast, merged results 205 can dramatically increase the efficiency and accuracy of debugging a design.

[0036] Figure 3 illustrates an exemplary flow in which the multiple runs are managed in series. In this flow, an STA tool 304 can receive a design and a generic STA set-up 309 as well as one or more potential set-ups 308 that specify modes and/or corners available for design analysis. STA tool 304 can then initiate multiple runs to be run in series using the available

set-ups. Once again, each run can be performed with a different mode and/or corner.

[0037] In this embodiment, an STA tool 302A can receive the design and an STA set-up 301A, perform the static timing analysis using a specified mode and corner on the design, and then save results 303A from that static timing analysis. An STA tool 302B can receive information 306A from run 300A as well as a design and STA set-up 301B, perform the static timing analysis using another specified mode and corner on the design, and then save results 303B from that analysis. Similarly, STA tool 302C can receive information 306B from run 300B as well as a design and STA set-up 301C, perform the static timing analysis using the specified mode and corner on the design, and then save results 303C.

[0038] Note that although this incremental flow can be slower than parallel analysis, some important efficiencies can be achieved. For example, in the parallel analysis shown in Figure 2, the netlist of the design is read for each run 200A, 200B, and 200C. In contrast, in a series analysis (also called an incremental flow), the netlist of the design can be read once in run 300A and then used for subsequent runs 300B and 300C. Therefore, information 306A and 306B could include a netlist of the design. Other information that can be shared between runs could include resistance/capacitance (e.g. parasitic network) information, timing graphs, etc. Therefore, in contrast to the independent runs of parallel analysis, the runs in an incremental flow can be related, i.e. share information. In one embodiment, STA tool 304 can automatically analyze the plurality of modes and corners to determine shared information between runs 300A, 300B, and 300C. This shared information can be provided in merged results 305.

[0039] Note that the information provided to downstream runs can include information generated in any previous run. For example, in one embodiment, information 306A can include at least saved results 303A, whereas information 306B can include at least saved results 303A as well as 303B. In this manner, STA tool 304 can merge saved results 303A, 303B, and 303C in merged results 305.

[0040] In accordance with one embodiment, STA tool 304 can perform pre-compilation, as appropriate. That is, if certain paths, modes, and/or corners are user-specified for analysis (i.e. reports are predictable), then such information can be provided separately from other data in the saved results. Advantageously, forming first merged results using separate reports for specified information can be done significantly faster than forming second merged results including all saved results (i.e. in batch mode). In other embodiments where user-specified information is not designated (i.e. reports are not predictable), certain default information (e.g. information typically requested by users) can be stored and organized. Once again, forming first merged results using separate reports for the default information could be done significantly faster than forming second merged results including all saved results. In yet other embodiments where specific information is not user-specified and default information is not designated, a user can query the merged results database after final compilation.

[0041] In accordance with one embodiment of the invention, a user can have the ability to interact on the fly with any number of runs. Specifically, the user can designate specific information (i.e. on an individual or group basis) for analysis at any time during the run(s). For example, if 20 runs were being simultaneously run and the user predicted that two runs having a certain corner would probably result in bad

performance, then the user could interact with that corner in those two runs to confirm or deny that prediction using a separate report in the merged results database. In one embodiment, a user can modify a predetermined set of parameters after completing an initial multi-mode/multi-corner analysis. At this point, an analysis can be performed by STA tool 304 to provide a what-if capability, thereby driving design optimization.

[0042] Although illustrative embodiments of the invention have been described in detail herein with reference to the figures, it is to be understood that the invention is not limited to those precise embodiments. They are not intended to be exhaustive or to limit the invention to the precise forms disclosed. For example, STA tool 204 (Figure 2) can automatically analyze the plurality of modes and corners to determine shared information between parallel runs 200A, 200B, and 200C. This shared information can be provided in merged results 205.

[0043] As such, many modifications and variations will be apparent. For example, a method of designing digital circuits could include performing a plurality of static timing analysis for different operation modes, manufacturing process corners, on operating conditions, or ambient environment conditions. The method could further include determining which scenarios should be simultaneously optimized and then optimizing selected scenarios simultaneously in logic synthesis, placement, and routing. Accordingly, it is intended that the scope of the invention be defined by the following claims and their equivalents.